上机3：Python数值分析

1. 开电脑

2. 打开PyCharm

授课老师：

胡晓敏（xmhu@ieee.org）

龚怡（2167570874@qq.com）

明俊峰（34940530@qq.com）

课程资料下载

- 课件

https://pan.baidu.com/s/1FuQaIoLmT1OnL1hAoBlkVQ

密码： she6

- Pycharm（或在Jetbrains官网下载，选择Community版本）

https://pan.baidu.com/s/1iXhXryPJG-YNYF-

上机3目录

- 理论教学：Python插值
- 实验：物体运动轨迹的插值预测

3.1 插值的应用

- 插值是离散函数逼近的重要方法
- 通过函数在有限个点处的取值状况，估算出函数在其他点处的近似值。
- 换句话说，插值法“模拟产生”一些新的但又比较靠谱的值来满足需求，这就是插值的作用。

3.1 插值的应用

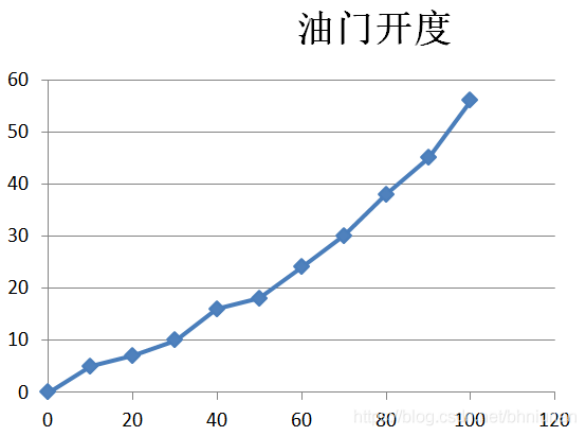
- 应用
  - 在人力资源管理中，应用插值法来设计一些考核规则。
    - 某公司的高管绩效考核规则：对其任务达成率进行考核。达成率大于150%时，其绩效得分为150分；达成率小于70%时，其绩效得分为0；达成率为70%~150%（包含70%与150%）时，按照实际达成率在60分至120分之间（包含60分与120分）利用线性插值法计算得分。

3.1 插值的应用

- 应用
  - 图像处理中，用来填充图像变换时像素之间的空隙。
    - 插值（Interpolation），有时也称为“重置样本”，是在不生成像素的情况下增加图像像素大小的一种方法，在周围像素色彩的基础上用数学公式计算丢失像素的色彩。简单地点，插值是根据中心像素点的颜色参数模拟出周边像素值的方法，是数码相机特有的放大数码照片的软件手段。

3.1 插值的应用

- 应用
  - 无人驾驶中的油门开度插值。
    - 在无人驾驶应用中，需要对车的驱动性能进行标定，需要标定出稳态下车速与油门踏板的对应关系。
    - 用线性插值就可以计算任何一个期望速度对应的问题期望油门踏板开度。



3.2 拉格朗日多项式插值

- 对给定的(x, y)数据data计算差商

```
from matplotlib import pyplot as plt

def Lg(data,testdata):
    predict=0
    data_x=[data[i][0] for i in range(len(data))]
    data_y=[data[i][1] for i in range(len(data))]
    if testdata in data_x:
        #print "testdata is already known"
        return data_y[data_x.index(testdata)]
    for i in range(len(data_x)):
        af=1
        for j in range(len(data_x)):
            if j!=i:
                af*=(1.0*(testdata-data_x[j])/(data_x[i]-data_x[j]))
        predict+=data_y[i]*af
    return predict
```

基函数

x = testdata对应的函数预测值

3.2 拉格朗日多项式插值

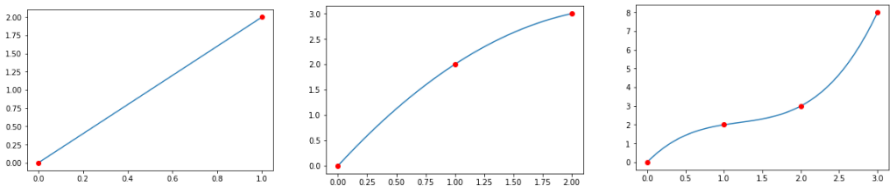
- 定义画曲线图的函数

```
def plot(data,nums):
    data_x=[data[i][0] for i in range(len(data))]
    data_y=[data[i][1] for i in range(len(data))]

    Area=[min(data_x),max(data_x)]

    X=[Area[0]+1.0*i*(Area[1]-Area[0])/nums for i in range(nums)]
    X[len(X)-1]=Area[1]

    Y=[Lg(data,x) for x in X]

    plt.plot(X,Y,label='result')

    for i in range(len(data_x)):
        plt.plot(data_x[i],data_y[i],'ro',label="point")

    plt.savefig('Lg.jpg')
    plt.show()
```

3.2 拉格朗日多项式插值

- 给定 $n+1$ 个点，得到次数不大于 n 次的多项式

```
#线性插值
data=[[0,0],[1,2]]
print(Lg(data,1.5))
plot(data,100)

#二次多项式插值
data=[[0,0],[1,2],[2,3]]
print(Lg(data,1.5))
plot(data,100)

#四次多项式插值
data=[[0,0],[1,2],[2,3],[3,8]]
print(Lg(data,1.5))
plot(data,100)

#五次多项式插值
data=[[0,0],[1,2],[2,3],[3,8],[4,2]]
print(Lg(data,1.5))
plot(data,100)

#六次多项式插值
data=[[0,0],[1,2],[2,3],[3,8],[4,2],[5,7]]
print(Lg(data,1.5))
plot(data,100)

#七次多项式插值
data=[[0,0],[1,2],[2,3],[3,8],[4,2],[5,7],[6,8]]
print(Lg(data,1.5))
plot(data,100)
```



1. 用表格比较不同次数的多项式对$x = 1.5$时得到的预测值
2. 画图整理插值曲线，分析它们的特点
3. 发现龙格现象吗？

3.3 牛顿多项式插值

- 对给定的(x, y)数据data计算差商

```
def calF(data):
    #差商计算  n个数据 0-(n-1)阶个差商 n个数据
    data_x=[data[i][0] for i in range(len(data))]
    data_y=[data[i][1] for i in range(len(data))]
    F= [1 for i in range(len(data))]
    FM=[]
    for i in range(len(data)):
        FME=[]
        if i==0:
            FME=data_y
        else:
            for j in range(len(FM[len(FM)-1])-1):
                delta=data_x[i+j]-data_x[j]
                value=1.0*(FM[len(FM)-1][j+1]-FM[len(FM)-1][j])/delta
                FME.append(value)
        FM.append(FME)
    F=[fme[0] for fme in FM]
    print(FM)
    return F
```

3.3 牛顿多项式插值

- 对给定的(x, y)数据，给定的预测位置，和差商F

```
def NT(data,testdata,F):
    #差商之类的计算
    predict=0
    data_x=[data[i][0] for i in range(len(data))]
    data_y=[data[i][1] for i in range(len(data))]
    if testdata in data_x:
        return data_y[data_x.index(testdata)]
    else:
        for i in range(len(data_x)):
            Eq=1
            if i!=0:
                for j in range(i):
                    Eq=Eq*(testdata-data_x[j])
                predict+=(F[i]*Eq)
    return predict
```

3.3 牛顿多项式插值

- 利用画曲线图的函数，计算插值函数和估计值

```
def plot(data,nums):
    data_x=[data[i][0] for i in range(len(data))]
    data_y=[data[i][1] for i in range(len(data))]

    Area=[min(data_x),max(data_x)]

    X=[Area[0]+1.0*i*(Area[1]-Area[0])/nums for i in range(nums)]
    X[len(X)-1]=Area[1]

    F=calF(data)        #计算差商
    Y=[NT(data,x,F) for x in X]    #牛顿插值

    plt.plot(X,Y,label='result')
    for i in range(len(data_x)):
        plt.plot(data_x[i],data_y[i],'ro',label="point")
    plt.savefig('Newton.jpg')
    plt.show()

data=[[0,0],[1,2],[2,3],[3,8],[4,2]]
plot(data,100)
```

3.4 分段线性插值

- 找出预测点所在的分段，计算插值函数和估计值

```
def DivideLine(data,testdata):
    #找到最邻近的
    data_x=[data[i][0] for i in range(len(data))]
    data_y=[data[i][1] for i in range(len(data))]

    if testdata in data_x:
        return data_y[data_x.index(testdata)]
    else:
        index=0
        for j in range(len(data_x)):
            if data_x[j]<testdata and  data_x[j+1]>testdata:
                index=j
                break
        predict=1.0*(testdata-data_x[j])*(data_y[j+1]-data_y[j])/(data_x[j+1]-data_x[j])+data_y[j]
        return predict
```

3.4 分段线性插值

- 画图

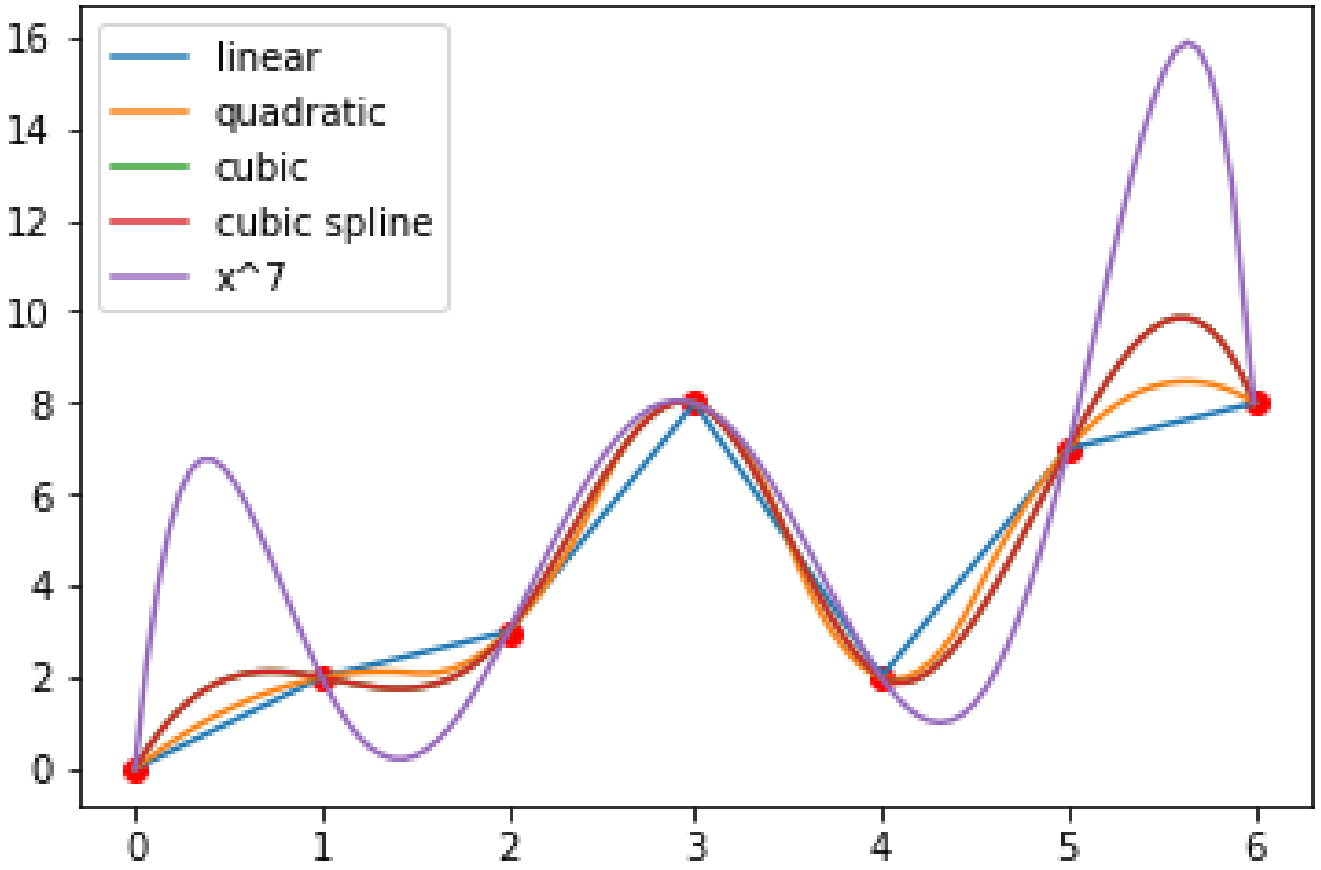
```
def plot(data,nums):
    data_x=[data[i][0] for i in range(len(data))]
    data_y=[data[i][1] for i in range(len(data))]

    Area=[min(data_x),max(data_x)]

    X=[Area[0]+1.0*i*(Area[1]-Area[0])/nums for i in range(nums)]
    X[len(X)-1]=Area[1]

    Y=[DivideLine(data,x) for x in X]

    plt.plot(X,Y,label='result')
    for i in range(len(data_x)):
        plt.plot(data_x[i],data_y[i],'ro',label="point")
    plt.savefig('DivLine.jpg')
    plt.show()

data=[[0,0],[1,2],[2,3],[3,8],[4,2]]
print(DivideLine(data,1.5))
plot(data,100)
```

如何修改为分段高次多项式插值？

3.5 分段插值的Python包

```
import numpy as np
from scipy import interpolate
import pylab as pl

data=[[0,0],[1,2],[2,3],[3,8],[4,2],[5,7],[6,8]]
x=[data[i][0] for i in range(len(data))]
y=[data[i][1] for i in range(len(data))]


xnew = np.linspace(0, 6, 101)  #用于画出插值曲线

pl.plot(x, y, "ro")

for kind in ["linear", "quadratic", "cubic"]:  # 插值方式1,2,3次多项式插值
    # linear 线性插值
    # quadratic 二次多项式插值
    # cubic 三次多项式插值
    f = interpolate.interp1d(x, y, kind=kind)  #选择对应方式的分段插值

    ynew = f(xnew)  #把 x 值代入插值函数，得到 y 坐标用于画出插值曲线
    pl.plot(xnew, ynew, label=str(kind))  # label用来显示图例

f4 = interpolate.splrep(x, y)  # 3次样条插值
ynew4 = interpolate.splev(xnew,f4,der=0)
pl.plot(xnew, ynew4, label=str("cubic spline"))  # label用来显示图例

pl.legend(loc="lower right")  # 显示图例的位置
pl.show()
```

比较不同插值方法的插值曲线



这个例子中，基于三次多项式的分段插值和三次样条插值曲线重合了

拓展：试一下用自己实现的分段高次多项式插值，看看曲线效果？

上机3目录

- 理论教学2：Python编程基础
- 实验：物体运动轨迹的插值预测

实验： 物体运动轨迹的插值预测

- **实验目的：**

  比较不同插值方法的预测精度。

- **实验背景：**

  在实际应用中，一般不能确知物体的运动轨迹方程，只能通过测量一些 x 坐标的值，通过数据插值的方法来求其他未知点的函数值。

实验： 物体运动轨迹的插值预测

- **已知测量的不同点的函数值如下：**

表 1　不同 x 点出物体的函数值

x	-5.0	-4.5	-4.0	-3.5	-3.0	-2.5	-2.0	-1.5	-1.0	-0.5
物体 1(y_1)	-0.1923	-0.2118	-0.2353	-0.2642	-0.3	-0.3448	-0.4000	-0.4615	-0.5000	-0.4000
物体 2(y_2)	0.0016	0.002	0.0025	0.0033	0.0044	0.0064	0.0099	0.0175	0.0385	0.1379

x	0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
物体 1(y_1)	0	0.4000	0.5000	0.4615	0.4000	0.3448	0.3000	0.2642	0.2353	0.2118	0.1923
物体 2(y_2)	1.0000	0.1379	0.0385	0.0175	0.0099	0.0064	0.0044	0.0033	0.0025	0.0020	0.0016

- **实验要求：**

采用拉格朗日插值、分段线性插值、样条插值等方法进行插值，并绘制插值函数的图形，根据结果，确定一种最好的插值方法。

实验： 物体运动轨迹的插值预测

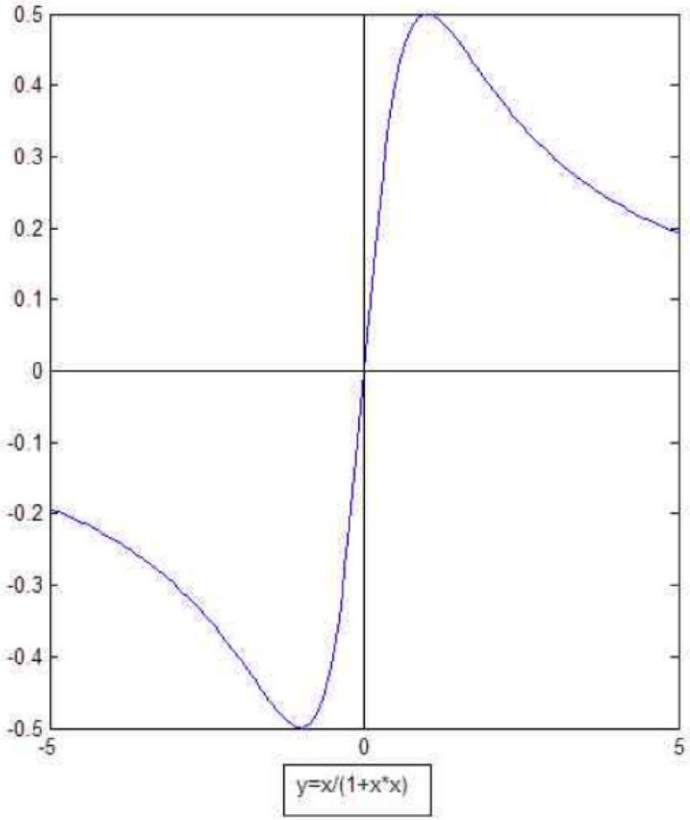
观察同一方案、不同插值节点数的计算结果的变化形态，不同方案结果的精度比较，有无龙格现象发生。

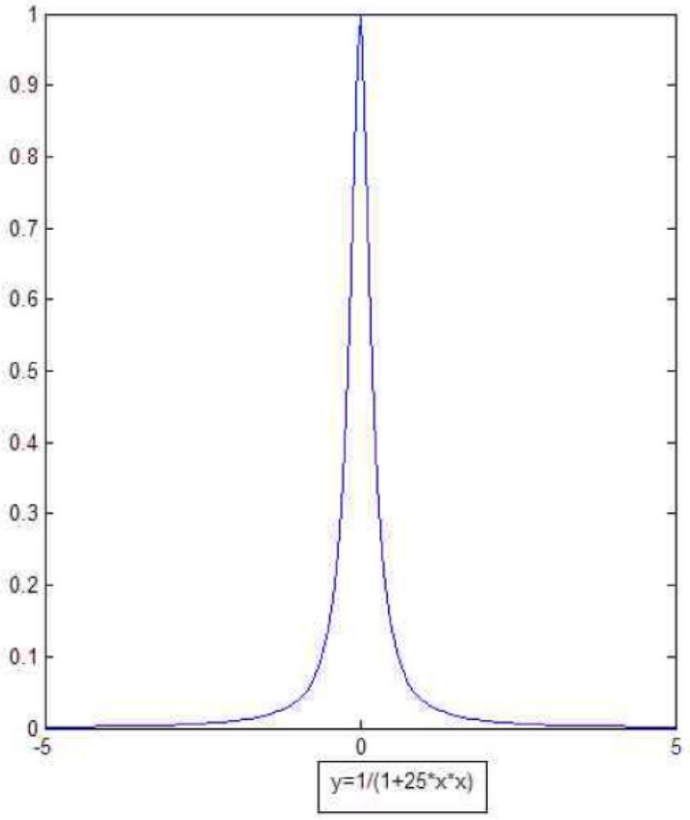
以上实验数据来源：

这个两个物体的移动满足

物体1： $y_1 = x/(1+x^2), \quad x \in [-5,5]$

物体2： $y_2 = 1/(1+25x^2), x \in [-5,5]$ **你插值的结果类似吗？**





上机课任务

- 完成实验内容，整理添加到课程报告内。

  并通过email发送给老师

word报告重命名为“专业班级学号姓名课程报告(含上机3).docx”

Email标题为“专业班级学号姓名课程报告(含上机3)”